This is a slightly revised version (July 2000) of the following article:

Marcin Jurdziński, Small Progress Measures for Solving Parity Games. In Horst Reichel and Sophie Tison, editors, *STACS 2000, 17th Annual Symposium on Theoretical Aspects of Computer Science, Proceedings, Lille, France, February 2000*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer.

Small Progress Measures for Solving Parity Games

Marcin Jurdziński[1]

BRICS[2]
Department of Computer Science
University of Aarhus

Abstract. In this paper we develop a new algorithm for deciding the winner in parity games, and hence also for the modal μ-calculus model checking. The design and analysis of the algorithm is based on a notion of game progress measures: they are witnesses for winning strategies in parity games. We characterize game progress measures as pre-fixed points of certain monotone operators on a complete lattice. As a result we get the existence of the least game progress measures and a straightforward way to compute them. The worst-case running time of our algorithm matches the best worst-case running time bounds known so far for the problem, achieved by the algorithms due to Browne et al., and Seidl. Our algorithm has better space complexity: it works in small polynomial space; the other two algorithms have exponential worst-case space complexity.

1 Introduction

A parity game is an infinite path-forming game played by two players, player $\Diamond$ and player $\Box$, on a graph with integer priorities assigned to vertices. In order to determine the winner in an infinite play we check the parity of the lowest priority occurring infinitely often in the play: if it is even then player $\Diamond$ wins, otherwise player $\Box$ is the winner. The problem of deciding the winner in parity games is, given a parity game and an initial vertex, to decide whether player $\Diamond$ has a winning strategy from the vertex.

There are at least two motivations for the study of the complexity of deciding the winner in parity games. One is that the problem is polynomial time equivalent to the modal μ-calculus model checking [3, 16], hence developing better algorithms for parity games may lead to better model checking tools, which is

[1] Address: BRICS, Department of Computer Science, University of Aarhus, Ny Munkegade, Building 540, 8000 Aarhus C, Denmark. Email: `mju@brics.dk`.

[2] **B**asic **R**esearch in **C**omputer **S**cience, Centre of the Danish National Research Foundation.

a major objective in computer aided verification. The other is that the problem has an interesting status from the point of view of structural complexity theory. It is known to be in **NP** $\cap$ **co-NP** [3] (and even in **UP** $\cap$ **co-UP** [6]), and hence it is very unlikely to be **NP**-complete, but at the same time it is not known to be in **P**, despite substantial effort of the community (see [3, 1, 15, 20] and references therein).

Progress measures [9] are decorations of graphs whose local consistency guarantees some global, often infinitary, properties of graphs. Progress measures have been used successfully for complementation of automata on infinite words and trees [7, 8]; they also underlie a translation of alternating parity automata on infinite words to weak alternating automata [10]. A similar notion, called a signature, occurs in the study of the modal μ-calculus [17]. Signatures have been used to prove memoryless determinacy of parity games [2, 18].

Our algorithm for parity games is based on the notion of game parity progress measures; Walukiewicz [18] calls them consistent signature assignments. Game parity progress measures are witnesses for winning strategies in parity games. We provide an upper bound on co-domains of progress measures; this reduces the search space of potential witnesses. Then we provide a characterization of game parity progress measures as pre-fixed points of certain monotone operators on a finite complete lattice. This characterization implies that the least game parity progress measures exist, and it also suggests an easy way to compute them.

The modal μ-calculus model checking problems is, given a formula φ of the modal μ-calculus and a Kripke structure K with a set of states S, to decide whether the formula is satisfied in the initial state of the Kripke structure. The problem has been studied by many researchers; see for example [5, 3, 1, 15, 11] and references therein. The algorithms with the best proven worst-case running time bounds so far are due to Browne et al. [1], and Seidl [15]. Their worst-case running times are roughly $O\big(m \cdot n^{\lceil d/2 \rceil}\big)$ and $O\big(m \cdot (n/d)^{\lceil d/2 \rceil}\big)$, respectively, where n and m are some numbers depending on φ and K, such that $n \leq |S| \cdot |\varphi|$, $m \leq |K| \cdot |\varphi|$, and d is the alternation depth of the formula φ.

In fact, number n above is the number of vertices in the parity game obtained from the formula and the Kripke structure via the standard reduction of the modal μ-calculus model checking to parity games, and m is the number of edges in the game graph; see for example [3, 16]. Moreover, the reduction can be done in such a way that the number of different priorities in the parity game is equal to the alternation depth d of the formula. Our algorithm has worst-case running time $O\big(m \cdot (n/\lfloor d/2 \rfloor)^{\lfloor d/2 \rfloor}\big)$, and it can be made to work in time $O\big(m \cdot \big((n+d)/d\big)^{\lceil d/2 \rceil}\big)$, hence it matches the bounds of the other two algorithms. Moreover, it works in space $O(dn)$ while the other two algorithms have exponential worst-case space complexity. Our algorithm can be seen as a generic algorithm allowing many different evaluation policies; good heuristics can potentially improve performance of the algorithm. However, we show a family of examples for which worst-case running time occurs for all evaluation policies.

Among algorithms for parity games it is worthwhile to mention the algorithm of McNaughton [12] and its modification due to Zielonka [19]. In the extended version of this paper we show that Zielonka's algorithm can be implemented to work in time roughly $O\big(m \cdot (n/d)^d\big)$, and we also provide a family of examples for which the algorithm needs this time. Zielonka's algorithm works in fact for games with more general Muller winning conditions. By a careful analysis of the algorithm for games with Rabin (Streett) winning conditions we get a running time bound $O\big(m \cdot n^{2k}/(k/2)^k\big)$, where k is the number of pairs in the Rabin (Streett) condition. The algorithm also works in small polynomial space. This compares favourably with other algorithms for the linear-time equivalent problem of checking non-emptiness of non-deterministic Rabin (Streett) tree automata [4, 14, 10], and makes it the best algorithm known for this **NP**-complete (co-**NP**-complete) [4] problem.

2 Parity games

Notation: For all $n \in \mathbb{N}$, by $[n]$ we denote the set $\{0, 1, 2, \dots, n-1\}$. If (V, E) is a directed graph and $W \subseteq V$, then by $(V, E) \restriction W$ we denote the subgraph (W, F) of (V, E), where $F = E \cap W^2$. [Notation] $\Box$

A *parity graph* $G = (V, E, p)$ consists of a directed graph (V, E) and a priority function $p : V \to [d]$, where $d \in \mathbb{N}$. A *parity game* $\Gamma = \big(V, E, p, (V_\Diamond, V_\Box)\big)$ consists of a parity graph $G = (V, E, p)$, called the *game graph* of Γ, and of a partition $(V_\Diamond, V_\Box)$ of the set of vertices V. For technical convenience we assume that all game graphs have the property that every vertex has at least one out-going edge. We also restrict ourselves throughout this paper to games with finite game graphs.

A parity game is played by two players: player $\Diamond$ and player $\Box$, who form an infinite path in the game graph by moving a token along edges. They start by placing the token on an initial vertex and then they take moves indefinitely in the following way. If the token is on a vertex in $V_\Diamond$ then player $\Diamond$ moves the token along one of the edges going out of the vertex. If the token is on a vertex in $V_\Box$ then player $\Box$ takes a move. In the result players form an infinite path $\pi = \langle v_1, v_2, v_3, \dots \rangle$ in the game graph; for brevity we refer to such infinite paths as *plays*. The winner in a play is determined by referring to priorities of vertices in the play. Let $\mathrm{Inf}(\pi)$ denote the set of priorities occurring infinitely often in $\big\langle p(v_1), p(v_2), p(v_3), \dots \big\rangle$. A play π is a *winning play* for player $\Diamond$ if $\min\big(\mathrm{Inf}(\pi)\big)$ is even, otherwise π is a winning play for player $\Box$.

A function $\sigma : V_\Diamond \to V$ is a *strategy* for player $\Diamond$ if $\big(v, \sigma(v)\big) \in E$ for all $v \in V_\Diamond$. A play $\pi = \langle v_1, v_2, v_3, \dots \rangle$ is *consistent* with a strategy σ for player $\Diamond$ if $v_{\ell+1} = \sigma(v_\ell)$, for all $\ell \in \mathbb{N}$, such that $v_\ell \in V_\Diamond$. A strategy σ is a *winning strategy* for player $\Diamond$ from set $W \subseteq V$, if every play starting from a vertex in W and consistent with σ is winning for player $\Diamond$. Strategies and winning strategies are defined similarly for player $\Box$.

Theorem 1 (Memoryless Determinacy [2, 13])
For every parity game, there is a unique partition $(W_\diamond, W_\square)$ *of the set of vertices of its game graph, such that there is a winning strategy for player* $\diamond$ *from* $W_\diamond$*, and a winning strategy for player* $\square$ *from* $W_\square$*.*

We call the sets $W_\diamond$ and $W_\square$ the *winning sets* of player $\diamond$ and player $\square$, respectively. The problem of *deciding the winner* in parity games is, given a parity game and a vertex in the game graph, to determine whether the vertex is in the winning set of player $\diamond$.

Before we proceed we mention a simple characterization of winning strategies for player $\diamond$ in terms of simple cycles in a subgraph of the game graph associated with the strategy. We say that a strategy σ for player $\diamond$ is *closed* on a set $W \subseteq V$ if for all $v \in W$, we have:

- if $v \in V_\diamond$ then $\sigma(v) \in W$, and
- if $v \in V_\square$ then $(v, w) \in E$ implies $w \in W$.

Note that if a strategy σ for player $\diamond$ is closed on W then every play starting from a vertex in W and consistent with σ stays within W.

If σ is a strategy for player $\diamond$ then by G_σ we denote the parity graph (V, E_σ, p) obtained from game graph $G = (V, E, p)$ by removing from E all edges (v, w) such that $v \in V_\diamond$ and $\sigma(v) \neq w$.

We say that a cycle in a parity graph is an *i-cycle* if i is the smallest priority of a vertex occurring in the cycle. A cycle is an even cycle if it is an i-cycle for some even i, otherwise it is an odd cycle. The following proposition is not hard to prove.

Proposition 2 Let σ be a strategy for player $\diamond$ closed on W. Then σ is a winning strategy for player $\diamond$ from W if and only if all simple cycles in $G_\sigma \upharpoonright W$ are even.

3 Small progress measures

In this section we study a notion of progress measures. Progress measures play a key role in the design and analysis of our algorithm for solving parity games.

First we define parity progress measures for parity graphs, and we show that there is a parity progress measure for a parity graph if and only if all cycles in the graph are even. In other words, parity progress measures are witnesses for the property of parity graphs having only even cycles. The proof of the 'if' part also provides an upper bound on the size of the co-domain of a parity progress measure. Then we define game parity progress measures for parity games, we argue that they are witnesses for winning strategies for player $\diamond$, and we show that the above-mentioned upper bound holds also for game parity progress measures.

Notation: If $\alpha \in \mathbb{N}^d$ is a d-tuple of non-negative integers then we number its components from 0 to $d-1$, i.e., we have $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{d-1})$. When applied to tuples of natural numbers, the comparison symbols $<, \leq, =, \neq, \geq$, and

$>$ denote the lexicographic ordering. When subscripted with a number $i \in \mathbb{N}$ (e.g., $<_i, =_i, \geq_i$), they denote the lexicographic ordering on $\mathbb{N}^i$ applied to the arguments truncated to their first i components. For example, $(2,3,0,0) >_2 (2,2,4,1)$, but $(2,3,0,0) =_0 (2,2,4,1)$. [Notation] □

Definition 3 (Parity progress measure)
Let $G = \big(V, E, p : V \to [d]\big)$ be a parity graph. A function $\varrho : V \to \mathbb{N}^d$ is a parity progress measure for G if for all $(v, w) \in E$, we have $\varrho(v) \geq_{p(v)} \varrho(w)$, and the inequality is strict if $p(v)$ is odd. [Definition 3] □

Proposition 4 If there is a parity progress measure for a parity graph G then all cycles in G are even.

Proof: Let $\varrho : V \to \mathbb{N}^d$ be a parity progress measure for G. For the sake of contradiction suppose that there is an odd cycle $v_1, v_2, \dots, v_\ell$ in G, and let $i = p(v_1)$ be the smallest priority on this cycle. Then by the definition of a progress measure we have $\varrho(v_1) >_i \varrho(v_2) \geq_i \varrho(v_2) \geq_i \cdots \geq_i \varrho(v_\ell) \geq_i \varrho(v_1)$, and hence $\varrho(v_1) >_i \varrho(v_1)$, a contradiction. [Proposition 4] ■

If $G = \big(V, E, p : V \to [d]\big)$ is a parity graph then for every $i \in [d]$, we write V_i to denote the set $p^{-1}(i)$ of vertices with priority i in parity graph G. Let $n_i = |V_i|$, for all $i \in [d]$. Define M_G to be the following finite subset of $\mathbb{N}^d$: if d is even then

$$M_G = [1] \times [n_1 + 1] \times [1] \times [n_3 + 1] \times \cdots \times [1] \times [n_{d-1} + 1];$$

for odd d we have $\cdots \times [n_{d-2}+1] \times [1]$ at the end. In other words, M_G is the finite set of d-tuples of integers with only zeros on even positions, and non-negative integers bounded by $|V_i|$ on every odd position i.

Theorem 5 (Small parity progress measure)
If all cycles in a parity graph G are even then there is a parity progress measure $\varrho : V \to M_G$ for G.

Proof: The proof goes by induction on the number of vertices in $G = \big(V, E, p : V \to [d]\big)$. For the induction to go through we slightly strengthen the statement of the theorem: we additionally claim, that if $p(v)$ is odd then $\varrho(v) >_{p(v)} (0, \dots, 0)$. The statement of the theorem holds trivially if G has only one vertex.

Without loss of generality we may assume that $V_0 \cup V_1 \neq \emptyset$; otherwise we can scale down the priority function of G by two, i.e., replace the priority function p by the function $p - 2$ defined by $\big(p-2\big)(v) = p(v) - 2$, for all $v \in V$. Suppose first that $V_0 \neq \emptyset$. By induction hypothesis there is a parity progress measure $\varrho : \big(V \setminus V_0\big) \to M_G$ for the subgraph $G \restriction \big(V \setminus V_0\big)$. Setting $\varrho(v) = (0, \dots, 0) \in M_G$, for all $v \in V_0$, we get a parity progress measure for G.

Suppose that $V_0 = \emptyset$ then $V_1 \neq \emptyset$. We claim that there is a non-trivial partition (W_1, W_2) of the set of vertices V, such that there is no edge from W_1 to W_2 in G.

Let $u \in V_1$; define $U \subseteq V$ to be the set of vertices to which there is a non-trivial path from u in G. If $U = \emptyset$ then $W_1 = \{u\}$ and $W_2 = V \setminus \{u\}$ is a desired partition of V. If $U \neq \emptyset$ then $W_1 = U$ and $W_2 = V \setminus U$ is a desired partition. The partition is non-trivial (i.e., $V \setminus U \neq \emptyset$) since $u \notin U$: otherwise a non-trivial path from u to itself gives a 1-cycle because $V_0 = \emptyset$, contradicting the assumption that all cycles in G are even.

Let $G_1 = G \restriction W_1$, and $G_2 = G \restriction W_2$ be subgraphs of G. By induction hypothesis there are parity progress measures $\varrho_1 : W_1 \to M_{G_1}$ for G_1, and $\varrho_2 : W_2 \to M_{G_2}$ for G_2. Let $n'_i = \left|V_i \cap W_1\right|$, and let $n''_i = \left|V_i \cap W_2\right|$, for $i \in [d]$. Clearly $n_i = n'_i + n''_i$, for all $i \in [d]$. Recall that there are no edges from W_1 to W_2 in G. From this and our additional claim applied to ϱ_2 it follows that the function $\varrho = \varrho_1 \cup \big(\varrho_2 + (0, n'_1, 0, n'_3, \dots)\big) : V \to M_G$ is a parity progress measure for G. [Theorem 5] ■

Let $\Gamma = \big(V, E, p, (V_\diamond, V_\Box)\big)$ be a parity game and let $G = (V, E, p)$ be its game graph. We define $M_G^\top$ to be the set $M_G \cup \{\top\}$, where $\top$ is an extra element. We use the standard comparison symbols (e.g., $<, =, \geq$, etc.) to denote the order on $M_G^\top$ which extends the lexicographic order on M_G by taking $\top$ as the biggest element, i.e., we have $m < \top$, for all $m \in M_G$. Moreover, for all $m \in M_G$ and $i \in [d]$, we set $m <_i \top$, and $\top =_i \top$. If $\varrho : V \to M_G^\top$ and $(v, w) \in E$ then

> by $\mathrm{Prog}(\varrho, v, w)$ we denote the least $m \in M_G^\top$, such that $m \geq_{p(v)} \varrho(w)$, and if $p(v)$ is odd then either the inequality is strict, or $m = \varrho(w) = \top$.

Definition 6 (Game parity progress measure)
A function $\varrho : V \to M_G^\top$ is a game parity progress measure if for all $v \in V$, we have:

- if $v \in V_\diamond$ then $\varrho(v) \geq_{p(v)} \mathrm{Prog}(\varrho, v, w)$ for some $(v, w) \in E$, and
- if $v \in V_\Box$ then $\varrho(v) \geq_{p(v)} \mathrm{Prog}(\varrho, v, w)$ for all $(v, w) \in E$;

by $\|\varrho\|$ we denote the set $\{\, v \in V \ : \ \varrho(v) \neq \top \,\}$. [Definition 6] □

For every game parity progress measure ϱ we define a strategy $\widetilde{\varrho} : V_\diamond \to V$ for player $\diamond$, by setting $\widetilde{\varrho}(v)$ to be a successor w of v, which minimizes $\varrho(w)$.

Corollary 7 If ϱ is a game parity progress measure then $\widetilde{\varrho}$ is a winning strategy for player $\diamond$ from $\|\varrho\|$.

Proof: Note first that ϱ restricted to $\|\varrho\|$ is a parity progress measure on $G_{\widetilde{\varrho}} \restriction \|\varrho\|$. Hence by Proposition 4 all simple cycles in $G_{\widetilde{\varrho}} \restriction \|\varrho\|$ are even.

It also follows easily from definition of a game parity progress measure that strategy $\widetilde{\varrho}$ is closed on $\|\varrho\|$. Therefore, by Proposition 2 we get that $\widetilde{\varrho}$ is a winning strategy for player $\diamond$ from $\|\varrho\|$. [Corollary 7] ■

Corollary 8 (Small game parity progress measure)
There is a game progress measure $\varrho : V \to M_G^\top$ such that $\|\varrho\|$ is the winning set of player $\diamond$.

Proof: It follows from Theorem 1 that there is a winning strategy σ for player $\Diamond$ from her winning set $W_\Diamond$, which is closed on $W_\Diamond$. Therefore by Proposition 2 all cycles in parity graph $G_\sigma \restriction W_\Diamond$ are even, hence by Theorem 5 there is a parity progress measure $\varrho : W_\Diamond \to M_G$ for $G_\sigma \restriction W_\Diamond$. It follows that setting $\varrho(v) = \top$ for all $v \in V \setminus W_\Diamond$, makes ϱ a game parity progress measure. [Corollary 8] ■

4 The algorithm

In this section we present a simple algorithm for solving parity games based on the notion of a game parity progress measure. We characterize game parity progress measures as (pre-)fixed points of certain monotone operators in a finite complete lattice. By Knaster-Tarski theorem it implies existence of the *least* game progress measure μ, and a simple way to compute it. It then follows from Corollaries 8 and 7 that $\|\mu\|$ is the winning set of player $\Diamond$.

Before we present the algorithm we define an ordering, and a family of $\mathrm{Lift}(\cdot, v)$ operators for all $v \in V$, on the set of functions $V \to M_G^\top$. Given two functions $\mu, \varrho : V \to M_G^\top$, we define $\mu \sqsubseteq \varrho$ to hold if $\mu(v) \leq \varrho(v)$ for all $v \in V$. The ordering relation $\sqsubseteq$ gives a complete lattice structure on the set of functions $V \to M_G^\top$. We write $\mu \sqsubset \varrho$ if $\mu \sqsubseteq \varrho$, and $\mu \neq \varrho$. Define $\mathrm{Lift}(\varrho, v)$ for $v \in V$ as follows:

$$\mathrm{Lift}\big(\varrho, v\big)(u) = \begin{cases} \varrho(u) & \text{if } u \neq v, \\ \max\big\{\varrho(v), \min_{(v,w)\in E} \mathrm{Prog}(\varrho, v, w)\big\} & \text{if } u = v \in V_\Diamond, \\ \max\big\{\varrho(v), \max_{(v,w)\in E} \mathrm{Prog}(\varrho, v, w)\big\} & \text{if } u = v \in V_\Box. \end{cases}$$

The following propositions follow immediately from definitions of a game parity progress measure, and of the $\mathrm{Lift}(\cdot, v)$ operators.

Proposition 9 For every $v \in V$, the operator $\mathrm{Lift}(\cdot, v)$ is $\sqsubseteq$-monotone.

Proposition 10 A function $\varrho : V \to M_G^\top$ is a game parity progress measure, if and only if is it is a simultaneous pre-fixed point of all $\mathrm{Lift}(\cdot, v)$ operators, i.e., if $\mathrm{Lift}(\varrho, v) \sqsubseteq \varrho$ for all $v \in V$.

From Knaster-Tarski theorem it follows that the $\sqsubseteq$-*least* game parity progress measure exists, and it can be obtained by running the following simple procedure computing the least simultaneous (pre-)fixed point of operators $\mathrm{Lift}(\cdot, v)$, for all $v \in V$.

```
ProgressMeasureLifting
    μ := λv ∈ V.(0, ..., 0)
    while μ ⊏ Lift(μ, v) for some v ∈ V do μ := Lift(μ, v)
```

Theorem 11 (The algorithm)
Given a parity game, procedure `ProgressMeasureLifting` *computes winning sets for both players and a winning strategy for player* $\Diamond$ *from her winning set; it works in space* $O(dn)$*, and its running time is*

$$O\Big(dm \cdot \Big(\frac{n}{\lfloor d/2 \rfloor}\Big)^{\lfloor d/2 \rfloor}\Big),$$

where n *is the number of vertices,* m *is the number of edges, and* d *is the maximum priority in the parity game.*

Proof: The result of running `ProgressMeasureLifting` on a parity game is the $\sqsubseteq$-least game progress measure μ. Let $W_\Diamond$ be the winning set of player $\Diamond$. By minimality of μ and by Corollary 8 it follows that $W_\Diamond \subseteq \|\mu\|$. Moreover, Corollary 7 implies that $\widetilde{\mu}$ is a winning strategy for player $\Diamond$ from $\|\mu\|$, and hence by Theorem 1 we get that $\|\mu\| \subseteq W_\Diamond$, i.e., $\|\mu\| = W_\Diamond$.

Procedure `ProgressMeasureLifting` algorithm works in space $O(dn)$ because it only needs to maintain a d-tuple of integers for every vertex in the game graph. The $\mathrm{Lift}(\cdot, v)$ operator, for every $v \in V$, can be implemented to work in time $O\big(d \cdot \text{out-deg}(v)\big)$, where out-deg$(v)$ is the out-degree of v. Every vertex can be "lifted" only $|M_G|$ many times, hence the running time of procedure `ProgressMeasureLifting` is bounded by

$$O\Big(\sum_{v \in V} d \cdot \text{out-deg}(v) \cdot |M_G|\Big) = O\big(d \cdot m \cdot |M_G|\big).$$

To get the claimed time bound it suffices to notice that

$$|M_G| = \prod_{i=1}^{\lfloor d/2 \rfloor} (n_{2i-1} + 1) \leq \Big(\frac{n}{\lfloor d/2 \rfloor}\Big)^{\lfloor d/2 \rfloor},$$

because $\sum_{i=1}^{\lfloor d/2 \rfloor}(n_{2i-1} + 1) \leq n$ if $n_i \neq 0$ for all $i \in [d]$, which we can assume without loss of generality; if $n_i = 0$ for some $i \in [d]$ then we can scale down the priorities bigger than i by two. [Theorem 11] ■

Remark: Our algorithm for solving parity games can be made to have

$$O\Big(d \cdot m \cdot \Big(\frac{n+d}{d}\Big)^{\lceil d/2 \rceil}\Big)$$

as its worst-case running time bound, which is better than $O\big(d{\cdot}m{\cdot}(n/\lfloor d/2 \rfloor)^{\lfloor d/2 \rfloor}\big)$ for even d, and for odd d if $2\log_{1/(1-\varepsilon)} n \leq d \leq (1-\varepsilon)n$, for some $0 < \varepsilon < 1$.

If $\sum_{0 \leq 2i-1 < d}(n_{2i-1} + 1) \leq (n+d)/2$ then we have

$$|M_G| \;=\; \prod_{0 \leq 2i-1 < d} (n_{2i-1} + 1) \;\leq\; \Big(\frac{(n+d)/2}{\lfloor d/2 \rfloor}\Big)^{\lfloor d/2 \rfloor} \;\leq\; \Big(\frac{n+d}{d-1}\Big)^{\lfloor d/2 \rfloor}.$$

Otherwise, we have $\sum_{0\le 2i<d}(n_{2i}+1) \le (n+d)/2$. In this case it suffices to run procedure `ProgressMeasureLifting` on the dual game $\overline{G}$, i.e., the game obtained by scaling all priorities in game G up by one, and swapping sets in the $(V_\diamond, V_\square)$ partition. The winning set of player $\diamond$ in the original game is the winning set of player $\square$ in the dual game. Note that

$$|M_{\overline{G}}| \;=\; \prod_{0\le 2i<d}(n_{2i}+1) \;\le\; \left(\frac{(n+d)/2}{\lceil d/2\rceil}\right)^{\lceil d/2\rceil} \;\le\; \left(\frac{n+d}{d}\right)^{\lceil d/2\rceil}.$$

To conclude it suffices to verify that $\big((n+d)/(d-1)\big)^{\lfloor d/2\rfloor} \le \big((n+d)/d\big)^{\lceil d/2\rceil}$, for $1 \le d \le n$. [Remark] $\square$

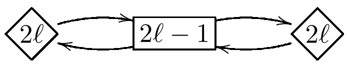
Note that in order to make `ProgressMeasureLifting` a fully deterministic algorithm one has to fix a policy of choosing vertices at which the function μ is being "lifted". Hence it can be considered as a generic algorithm whose performance might possibly depend on supplying heuristics for choosing the vertices to lift. Unfortunately, as we show in the next section, there is a family of examples on which the worst case performance of the algorithm occurs for all vertex lifting policies.

5 Worst-case behaviour

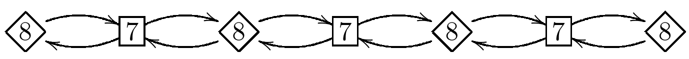
Theorem 12 (Worst-case behaviour)
For all $d, n \in \mathbb{N}$ such that $d \le n$, there is a game of size $O(n)$ with priorities not bigger than d, on which procedure `ProgressMeasureLifting` *performs at least $\big(\lceil n/d\rceil\big)^{\lceil d/2\rceil}$ many lifts, for all lifting policies.*

Proof: We define the family of games $H_{\ell,b}$, for all $\ell, b \in \mathbb{N}$. The game graph of $H_{\ell,b}$ consists of ℓ "levels", each level contains b "blocks". There is one "odd" level, and $\ell - 1$ "even" levels.

The basic building block of the odd level is the following subgraph.

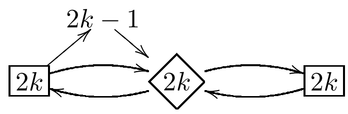


The numbers in vertices are their priorities. The odd level of $H_{\ell,b}$ consists of b copies of the above block assembled together by identifying the left-hand vertex with priority 2ℓ of the a-th block, for every $a \in \{1, 2, \ldots, b-1\}$, with the right-hand vertex with priority 2ℓ of the $(a+1)$-st block. For example the odd level of $H_{4,3}$ is the following.



In all our pictures vertices with a diamond-shaped frame are meant to belong to $V_\diamond$, i.e., they are vertices where player $\diamond$ moves; vertices with a box-shaped frame belong to $V_\Box$. Some vertices have no frame; for concreteness let us assume that they belong to $V_\Diamond$, but including them to $V_\Box$ would not change our reasoning, because they all have only one successor in the game graph of $H_{\ell,b}$.

The basic building block of the k-th even level, for $k \in \{1, 2, \ldots, \ell - 1\}$, is the following subgraph.



Every even level is built by putting b copies of the above block together in a similar way as for the odd level.

To assemble the game graph of $H_{\ell,b}$ we connect all $\ell - 1$ even levels to the odd level, by introducing edges in the following way. For every even level $k \in \{1, 2, \ldots, \ell - 1\}$, and for every block $a \in \{1, 2, \ldots, b\}$, we introduce edges in both directions between the box vertex with priority $2\ell - 1$ from the a-th block of the odd level, and the diamond vertex with priority $2k$ from the a-th block of the k-th even level. See Figure 1 for an example: the game $H_{4,3}$.

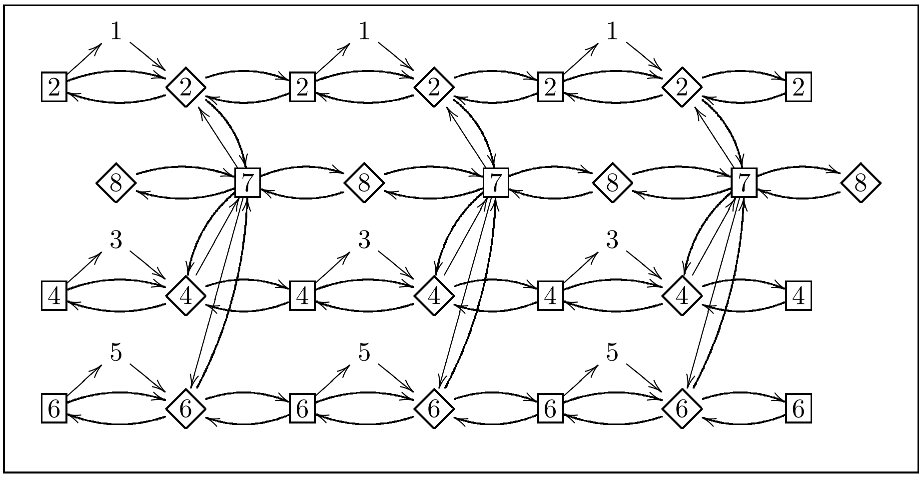


Fig. 1. The game $H_{4,3}$.

Claim 13 Every vertex with priority $2\ell - 1$ in game $H_{\ell,b}$ is lifted $(b+1)^\ell$ many times by procedure `ProgressMeasureLifting`.

Proof: Note that in game $H_{\ell,b}$ player $\Diamond$ has a winning strategy from all vertices in even levels, and player $\Box$ has a winning strategy from all vertices in the odd level; see Figure 1. Therefore, the value of the least progress measure in all vertices with priority $2\ell - 1$ is $\top \in M^{\top}_{H_{\ell,b}}$. Hence it suffices to show that every vertex with priority $2\ell - 1$ can be lifted only to its immediate successor in the order on $M^{\top}_{H_{\ell,b}}$. Then it is lifted $\left|M_{H_{\ell,b}}\right| = (b+1)^{\ell}$ many times, because

$$M_{H_{\ell,b}} = \underbrace{[1] \times [b+1] \times [1] \times [b+1] \times \cdots \times [b+1] \times [1]}_{2\ell+1 \text{ components}}.$$

Let v be a vertex with priority $2\ell - 1$ in the odd level of $H_{\ell,b}$, and let w be a vertex, such that there is an edge from v to w in the game graph of $H_{\ell,b}$. Then there is also an edge from w to v in the game graph of $H_{\ell,b}$; see Figure 1. Therefore, function μ maintained by the algorithm satisfies $\mu(w) \leq \mu(v)$, because w is a diamond vertex with even priority, so $\mathrm{Prog}(\mu, w, v) =_{p(w)} \mu(v)$, and $\big(\mathrm{Prog}(\mu, w, v)\big)_i = 0$ for all $i > p(w)$. It follows that $\mathrm{Lift}(\cdot, v)$ operator can only lift $\mu(v)$ to the immediate successor of $\mu(v)$ in the order on $M_{H_{\ell,b}}$, because the priority of v is $2\ell - 1$. [Claim 13] ■

Theorem 12 follows from the above claim by taking the game $H_{\lfloor d/2 \rfloor, \lceil n/d \rceil}$.

[Theorem 12] ■

6 Optimizations

Even though procedure `ProgressMeasureLifting` as presented above admits the worst-case performance, there is some room for improvements in its running time. Let us just mention here two proposals for optimizations, which should be considered when implementing the algorithm.

One way is to get better upper bounds on the values of the least game parity progress measure than the one provided by Corollary 8, taking into account the structure of the game graph. This would allow to further reduce the "search space" where the algorithm is looking for game progress measures. For example, let $G^{\geq i}$ be the parity graph obtained from the game graph G by removing all vertices with priorities smaller than i. One can show that if $v \in \|\mu\|$ for the least game progress measure μ then for odd i's the i-th component of $\mu(v)$ is bounded by the number of vertices of priority i reachable from v in graph $G^{\geq i}$. It requires further study to see whether one can get considerable improvements by pre-computing better bounds for the values of the least game parity progress measure.

Another simple but important optmization is to decompose game graphs into maximal strongly connected components. Note that every infinite play eventually stays within a strongly connected component, so it suffices to apply expensive procedure for solving parity games to the maximal strongly connected components separately. In fact, we need to proceed bottom up in the partial order of maximal strongly connected components. Each time one of the bottom components has been solved, we can also remove from the rest of the game the sets of vertices from which respective players have a strategy to force in a finite number of moves to their so far computed winning sets.

The above optimizations should considerably improve performance of our algorithm in practice, but they do not, as such, give any asymptotic worst-case improvement: see the examples $H_{\ell,b}$ from Section 5.

Acknowledgements

I am indebted to Mogens Nielsen, Damian Niwiński, Igor Walukiewicz, and Jens Vöge for numerous inspiring discussions on the subject. I thank anonymous STACS 2000 referees for very helpful advice on improving the focus and presentation of the paper.

References

1. A. Browne, E. M. Clarke, S. Jha, D. E. Long, and W. Marrero. An improved algorithm for the evaluation of fixpoint expressions. *Theoretical Computer Science*, 178(1–2):237–255, May 1997.
2. E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy (Extended abstract). In *Proceedings of 32nd Annual Symposium on Foundations of Computer Science*, pages 368–377. IEEE Computer Society Press, 1991.
3. E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model-checking for fragments of μ-calculus. In Costas Courcoubetis, editor, *Computer Aided Verification, 5th International Conference, CAV'93*, volume 697 of *LNCS*, pages 385–396, Elounda, Greece, June/July 1993. Springer-Verlag.
4. E. Allen Emerson and Charanjit S. Jutla. The complexity of tree automata and logics of programs. In *Proceedings of 29th Annual Symposium on Foundations of Computer Science*, pages 328–337, White Plains, New York, 24–26 October 1988. IEEE Computer Society Press.
5. E. Allen Emerson and Chin-Laung Lei. Efficient model checking in fragments of the propositional mu-calculus (Extended abstract). In *Proceedings, Symposium on Logic in Computer Science*, pages 267–278, Cambridge, Massachusetts, 16–18 June 1986. IEEE.
6. Marcin Jurdziński. Deciding the winner in parity games is in UP $\cap$ co-UP. *Information Processing Letters*, 68(3):119–124, November 1998.
7. Nils Klarlund. Progress measures for complementation of ω-automata with applications to temporal logic. In *32nd Annual Symposium on Foundations of Computer Science*, pages 358–367, San Juan, Puerto Rico, 1–4 October 1991. IEEE.
8. Nils Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Annals of Pure and Applied Logic*, 69(2–3):243–268, 1994.
9. Nils Klarlund and Dexter Kozen. Rabin measures and their applications to fairness and automata theory. In *Proceedings, Sixth Annual IEEE Symposium on Logic in Computer Science*, pages 256–265, Amsterdam, The Netherlands, 15–18 July 1991. IEEE Computer Society Press.
10. Orna Kupferman and Moshe Y. Vardi. Weak alternating automata and tree automata emptiness. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pages 224–233, Dallas, Texas, USA, 23–26 May 1998. ACM Press.
11. Xinxin Liu, C. R. Ramakrishnan, and Scott A. Smolka. Fully local and efficient evaluation of alternating fixed points. In Bernhard Steffen, editor, *Tools and Algorithms for Construction and Analysis of Systems, 4th International Conference, TACAS '98*, volume 1384 of *LNCS*, pages 5–19, Lisbon, Portugal, 28 March–4 April 1998. Springer.
12. Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149–184, 1993.

13. A. W. Mostowski. Games with forbidden positions. Technical Report 78, University of Gdańsk, 1991.
14. Amir Pnueli and Roni Rosner. On the synthesis of a reactive module. In *Conference Record of the 16th Annual ACM Symposium on Principles of Programming Languages (POPL '89)*, pages 179–190, Austin, Texas, January 1989. ACM Press.
15. Helmut Seidl. Fast and simple nested fixpoints. *Information Processing Letters*, 59(6):303–308, September 1996.
16. Colin Stirling. Local model checking games (Extended abstract). In Insup Lee and Scott A. Smolka, editors, *CONCUR'95: Concurrency Theory, 6th International Conference*, volume 962 of *LNCS*, pages 1–11, Philadelphia, Pennsylvania, 21–24 August 1995. Springer-Verlag.
17. Robert S. Streett and E. Allen Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Information and Computation*, 81(3):249–264, 1989.
18. Igor Walukiewicz. Pushdown processes: Games and model checking. In Thomas A. Henzinger and Rajeev Alur, editors, *Computer Aided Verification, 8th International Conference, CAV'96*, volume 1102 of *LNCS*, pages 62–74. Springer-Verlag, 1996. Full version available through `http://zls.mimuw.edu.pl/~igw`.
19. Wiesław Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200:135–183, 1998.
20. Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158:343–359, 1996.